

UD X - INTRODUCCIÓN
A JAVASCRIPT COMO
LENGUAJE DE LA WEB

The JavaScript logo, consisting of the letters 'JS' in a bold, dark blue font, centered within a large yellow circle. The background of the slide features abstract geometric shapes in shades of green, purple, and pink.

JS

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

Índice de Contenidos

- ▶ 1 - Introducción
- ▶ 2 - De Java a JavaScript
- ▶ 3 - Entorno de trabajo y primer programa
- ▶ 4 - Elementos básicos del lenguaje
- ▶ 5 - Mezclando JS con HTML
- ▶ 6 - Introducción a la manipulación web con JS

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

1 - Introducción

1 - Introducción

- ▶ Un poco de historia. Mediados de los 90. Internet entra en los hogares y tiene una gran proyección de mercado. Las páginas web son "feas", CSS está naciendo y HTML en la versión 2.0.
- ▶ Es la época de la "guerra de los navegadores". Internet Explorer y Netscape luchan por ofrecer la mejor experiencia de navegación al usuario para ganar cuota de mercado.
- ▶ Java quiere su parte del pastel y crea los "applets" una tecnología que permite ejecutar programas en el navegador pero es lenta y requiere instalar plugins en el navegador... ³



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

1 - Introducción

- ▶ JavaScript se desarrolla en Netscape a finales de 1995 para dotar de **dinamismo** y de **interactividad** a las páginas web de la época.
 - ▶ Inicialmente se iba a llamar LiveScript pero se quiso aprovechar el tirón mediático de Java y se le llamó JavaScript.
- ▶ Se **crea en tiempo "record"** (10 días) y se pretende que el lenguaje sea **fácil de aprender** para "todo el mundo". Estas dos circunstancias hacen que sea un lenguaje bastante peculiar y que arrastra errores de diseño desde sus inicios.
 - ▶ Tiene el "honor" de ser simultáneamente el lenguaje más amado y odiado por la comunidad de desarrolladores.
 - ▶ Aprender JS implica también aprender a quedarte las "partes buenas" del lenguaje y esquivar las "partes malas"...



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

1 - Introducción

JS vs ECMAScript

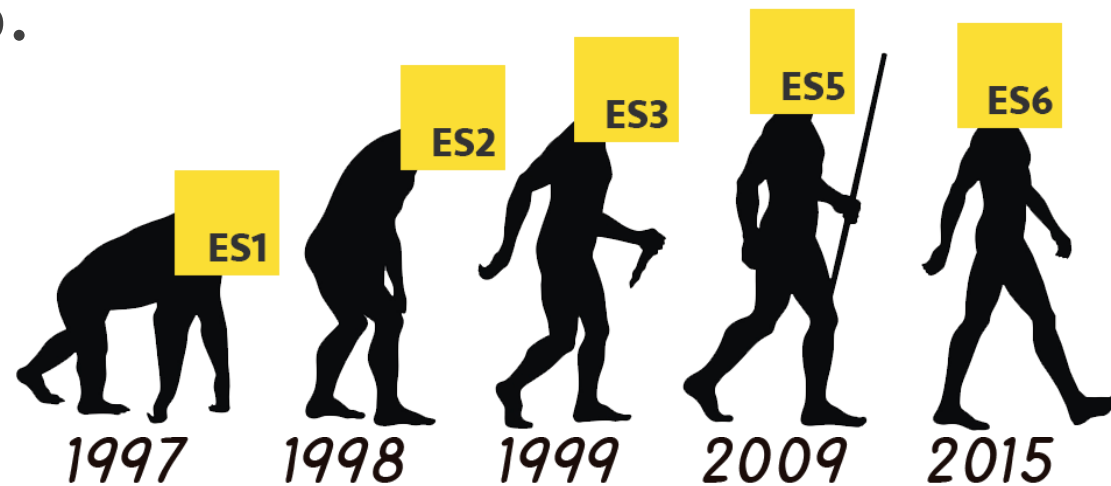
- ▶ JS va ganando fama y los distintos navegadores desarrollan sus propios intérpretes del lenguaje. Esto provoca "**divergencias**" y un mismo código se puede comportar de forma distinta dependiendo del navegador que lo ejecute.
- ▶ La ECMA (European Computer Manufacturers Association) decide intervenir y crear **ECMAScript** que es una especificación de cómo debe interpretarse el código JS.
- ▶ Técnicamente JavaScript y ECMAScript son dos cosas distintas, pero coloquialmente son términos intercambiables.



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

1 - Introducción

- Las especificaciones de ECMAScript (ES) han ido pasando por distintas versiones. La actual es la ES11 aunque la que está integrada en el 100% de los navegadores modernos a día de hoy es la ES6.



- En esta unidad estudiaremos el JavaScript de la especificación ES6 que es la más popular actualmente.

JS
{ES6}

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

2 - De Java a JavaScript

2 - De Java a JavaScript

- ▶ Java y Javascript (JS) son dos lenguajes DISTINTOS, solo que por razones históricas/comerciales, ambos comienzan por la palabra "Java".
- ▶ Eso significa que para aprender JS tendremos que "abandonar" ciertos conceptos que funcionan en Java y no "forzar" que un lenguaje encaje en el otro.
- ▶ Antes de entrar en los detalles de JS, hagamos una comparativa de ambos lenguajes para ver comprender mejor sus diferencias.



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

2 - De Java a JavaScript

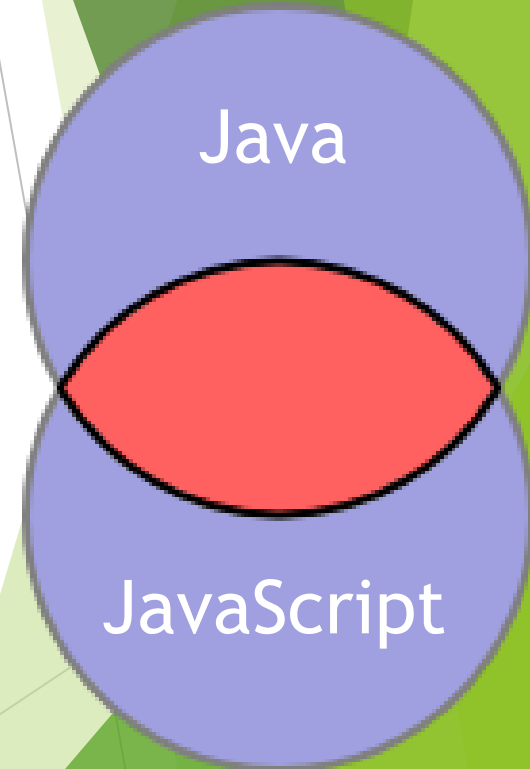
► Comparativa:

JAVA	JAVASCRIPT
<ul style="list-style-type: none">• Sigue el paradigma Orientado a Objetos puro.• Existen modificadores de acceso: private, public...• No permite el paradigma Estructurado. Los métodos siempre se presentan dentro de una clase.• No es un lenguaje de script y para ejecutar un programa hace falta una estructura concreta: método main.• Es de tipado fuerte (si una variable es de un tipo será de ese tipo durante todo el programa)• Se compila a un código intermedio que posteriormente es interpretado por la MVJ.• El compilador es "estricto" con el programador y no hace suposiciones, si hay un error te lo señala.• Su entorno de ejecución es la MVJ que puede "estar sola" o embebida en un navegador.	<ul style="list-style-type: none">• Sigue un paradigma OO "propio" que no está basado en clases, sino en prototipos.• No existen modificadores de acceso a los miembros de un objeto. Casi todo se considera public.• Permite el paradigma Estructurado en el que las funciones son métodos "sueños", que no están incluidos en ninguna clase.• Es un lenguaje de script, así que para lanzar un script solo hace falta una sucesión de instrucciones, sin apenas estructura.• Es de tipado débil (las variables pueden cambiar de tipo durante el programa)• No se compila, el código fuente se interpreta directamente en un entorno de ejecución.• El intérprete es "amable" con el programador y hace algunas suposiciones para "pasar por alto" algunos errores.• Su entorno de ejecución "natural" es el intérprete de JS que tienen los navegadores, aunque también hay intérpretes "en solitario" como Node.JS.

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

2 - De Java a JavaScript

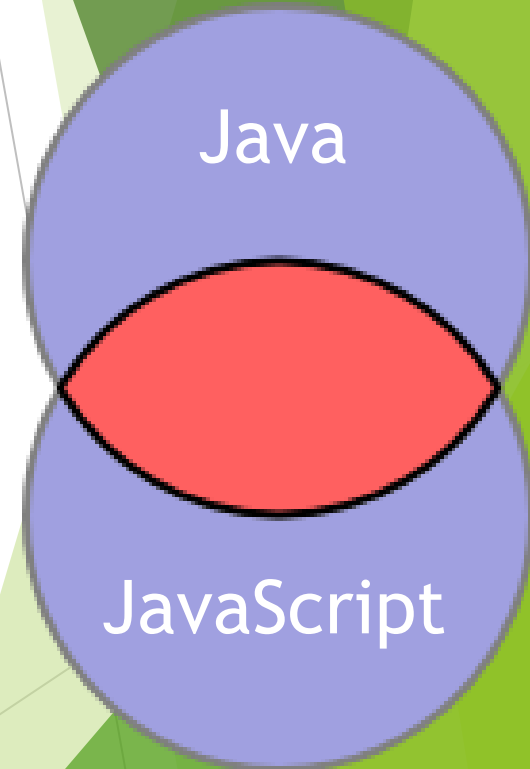
- ▶ Pero también hay muchos conceptos y **elementos sintácticos comunes** que funcionan igual en ambos lenguajes:
 - ▶ Comentarios con `//` o con `/* ... */`
 - ▶ Las instrucciones terminan con `;` (aunque los navegadores aceptan que se te olvide)
 - ▶ Se consideran identificadores válidos (para nombres de funciones o variables...) aquellos que comienza por letra, `_` ó `$` y le siguen otras letras, números ó `_` ó `$`
 - ▶ El estilo de nombrado también sigue las mismas reglas que en Java (UpperCamelCase para Clases/Prototipos y lowerCamelCase para el resto de elementos del lenguaje)



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

2 - De Java a JavaScript

- ▶ Mismas reglas de indentación
- ▶ Condicionales: if, if...else y switch
- ▶ Condicional ternaria: `condicion ? valor1 : valor2;`
- ▶ Bucles: while, do...while y for
- ▶ Operadores de asignación: `=`, `+=`, `-=` ...
- ▶ Operadores aritméticos: `+`, `-`, `*`, `/`, `%`...
- ▶ Operadores de autoincremento/decremento: `++`, `--`
- ▶ Operadores de comparación: `==`, `!=`, `>=`, `<=`... (aunque el `==` funciona un poco diferente). Además, Javascript añade otro operador de comparación extra: `===`



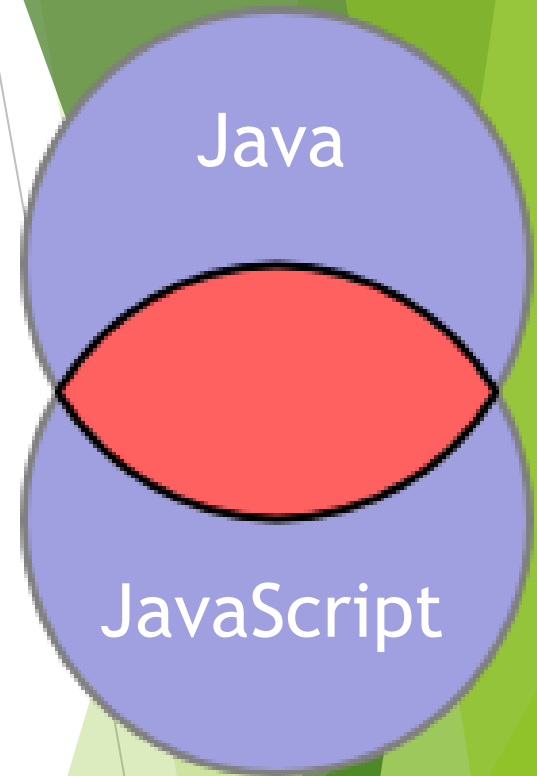
UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

2 - De Java a JavaScript

- ▶ Operadores lógicos: &&, || y !
- ▶ Operador de concatenación de cadenas: +
- ▶ Operador de acceso a un miembro de un objeto: .
- ▶ Orden de prioridad de los operadores
- ▶ Invocación de funciones/métodos:

`nombreDeLaFuncion(argumento1, argumento2)`

Eso sí... *"no es **ORO** todo lo que reluce"*. La forma de trabajar con los objetos y prototipos de JS es MUY DIFERENTE a cómo se realiza en Java... Ya lo estudiaréis en 2º curso...



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

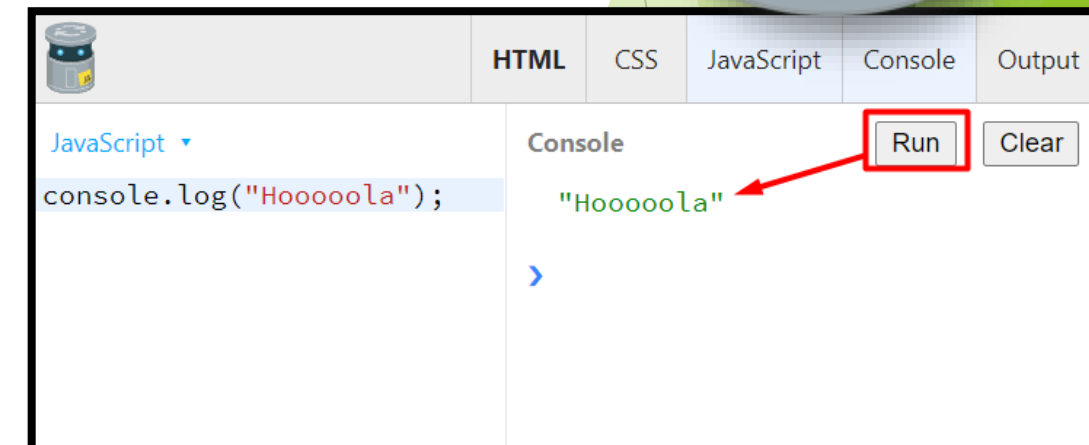
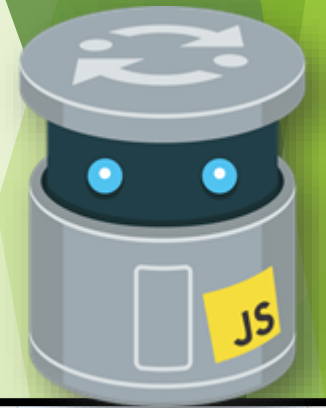
3 - Entornos de desarrollo y primer programa

3 - Entornos de desarrollo y primer programa

- ▶ En apartado 6 de esta unidad integraremos JS con HTML y CSS y utilizaremos la pareja VSCode-Navegador como entorno de desarrollo.
- ▶ Sin embargo, en esta primera parte, veremos un código JS independiente de HTML/CSS. Así que utilizaremos uno de los **entornos web de experimentación** disponibles en Internet que nos permitirán ver simultáneamente el código y la consola.

- ▶ JS Bin (papelera JS) tiene esta pinta 

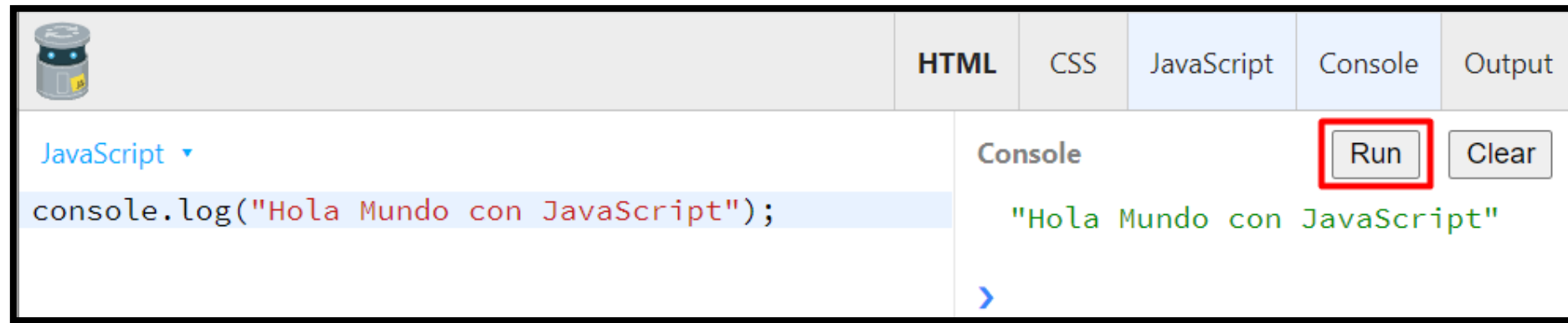
- ▶ Se accede desde la web: <https://jsbin.com/?js,console>
- ▶ Usaremos las pestañas "JavaScript" y "Console".
- ▶ Con el botón "Run" ejecutamos el script de JS.
- ▶ Con el botón "Clear" se limpian los mensajes de la consola.



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

3 - Entornos de desarrollo y primer programa

- Nuestro primer programa simplemente escribirá un mensaje en la consola.



- ¡Ea! Ya sabemos escribir en la consola de JS. **Console** es un objeto que definen todos los navegadores y **log** es un método de este objeto.
- Como vemos hemos lanzado un **script**, es decir, una secuencia de instrucciones. No tenemos un **main**, ni estamos obligados a declarar clases, ni debemos seguir una estructura concreta.



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

4 - Elementos básicos del lenguaje

Declaración de variables y constantes

- Observa el siguiente script y su resultado en consola.

```
JavaScript ▾  
let edad = 33;  
let nombre = "Pepe";  
console.log("Hola "+nombre+" tienes "+edad+" años");
```

Console Run Clear

"Hola Pepe tienes 33 años"

- Como vemos las variables se declaran con **let** ("permitir" que se use la variable...) **Y NO HAY QUE PONER EL TIPO**. El tipo se "deduce" directamente de la asignación.

¿Y qué pasa si no le asigno nada a la variable?



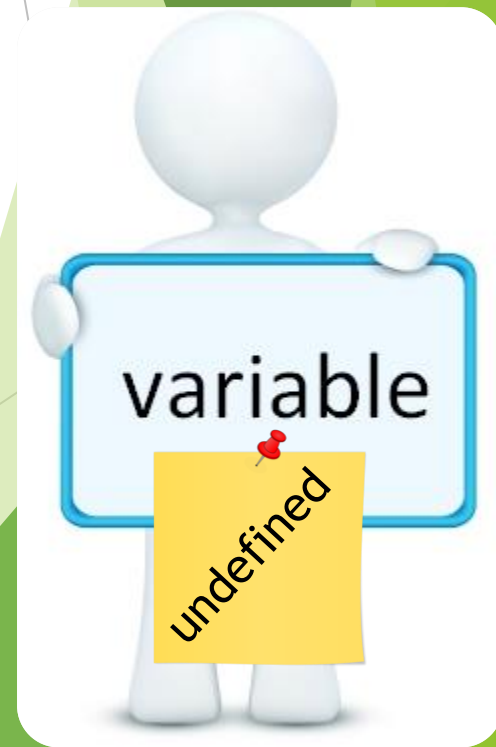
UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

JavaScript ▾ <pre>let edad; let nombre = "Pepe"; console.log("Hola "+nombre+" tienes "+edad+" años");</pre>	Console "Hola Pepe tienes undefined años"
--	--

- ▶ Como vemos las variables no inicializadas están "marcadas" como "undefined" y así aparecerán cada vez que queramos mostrarlas en la consola.
- ▶ En cuanto se les asigna un valor, se les quita esta marca.

JavaScript ▾ <pre>let edad; let nombre = "Pepe"; console.log("Hola "+nombre+" tienes "+edad+" años"); edad = 21; console.log("Hola "+nombre+" tienes "+edad+" años");</pre>	Console "Hola Pepe tienes undefined años" "Hola Pepe tienes 21 años"
--	--



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- Las constantes se definen con **const** y una vez se les asigne un valor no se podrá cambiar.

JavaScript	Console
<pre>const PI = 3.1416; let radio = 10; let perim = 2*PI*radio; console.log("El perímetro es: "+perim); // Esto no está permitido PI = 7;</pre>	<pre>"El perímetro es: 62.832" "error" "TypeError: Assignment to constant variable." at soqimatoru.js:7:4 at https://static.jsbin.com/js/prod/runner-4.1.8.min.js:1:13924 at https://static.jsbin.com/js/prod/runner-4.1.8.min.js:1:10866"</pre>

Realiza los ejercicios 1 y 2 del boletín de problemas

- Hay otra forma de declarar variables que usa **var** en vez de **let** pero hoy se considera una mala práctica (da problemas de "ámbito")

var edad = 33;



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

Los tipos de datos en JS

- ▶ Al igual que en Java, JS clasifica los tipos de datos en dos grandes grupos:
 - ▶ **Tipos primitivos:** son tipos simples que sirven para representar un valor concreto de una magnitud (3, true, 4.5...). No son objetos, no tienen métodos ni propiedades.
 - ▶ Los estudiamos en el siguiente diapositiva.
 - ▶ **Tipos objeto:** representan una realidad más compleja que normalmente forma parte del problema que se quiere resolver (Factura, Nómina, Persona...). Poseen propiedades y métodos. Al igual que en Java, se usan referencias para "apuntar" a los objetos que se creen en JS.
 - ▶ La creación y utilización de objetos en JS se estudia en el 2º curso de este ciclo.



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

Los tipos primitivos en JS son:

- ▶ **number**: es el equivalente al double en Java. Es un número real que se almacena en 8 bytes. En JS no existe ni int ni integer. **Todos los números son number.**
- ▶ **string**: sirve para representar textos con caracteres Unicode de 16 bits. En JS no existe el tipo char. Si queremos representar un solo carácter usaremos una cadena para ello. String en Java es un objeto pero string en JS no.
- ▶ **boolean**: define los valores true y false. Funciona igual que en Java.
- ▶ **undefined**: es un tipo nuevo que define un solo valor llamado "undefined". Como ya vimos, este valor se usa para "marcar" a una variable que aún no se le ha asignado ningún valor.
- ▶ **null**: es un tipo que solo define el valor "null" y se usa para "marcar" que una referencia a un objeto aún no ha sido inicializada.



Observa que todos empiezan con minúscula ya que **NO SON OBJETOS**

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

JS es un lenguaje de tipado débil

Cuando hablamos del "**tipado**" de un lenguaje nos referimos a la forma en la que gestiona el tipo de las variables y expresiones.

Java es un lenguaje de "**tipado fuerte**" porque nos obliga a definir el tipo de una variable cuando la declaramos, entonces **el tipo queda "fijado"** durante todo el programa. Además, no podemos hacer operaciones que mezclen distintos tipos.

Por ejemplo, esto en Java se considera un error porque la variable *mayorEdad* está declarada como boolean así que no puede almacenar un entero:

```
boolean mayorEdad = true;  
mayorEdad = 18;
```



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

JS es un lenguaje de "tipado débil" y se permite que las variables puedan cambiar su tipo durante la ejecución del programa. Observa:

Tipo number	<pre>JavaScript ▾ let mayorEdad = true; console.log ("Valor de la variable = "+mayorEdad); mayorEdad = 18; console.log ("Valor de la variable = "+mayorEdad);</pre>	Console
		<pre>"Valor de la variable = true" "Valor de la variable = 18"</pre>

Con este nuevo enfoque las variables no tienen un tipo fijo o bien que ahora son de tipo "mixto". Solo podremos afirmar que en un instante dado una variable contiene un valor cuyo tipo es X. Esto permite una sintaxis más libre y expresiva.

Sin embargo, JS además hace "conversiones automáticas de tipo" en cualquier tipo de expresión. Esto abre una "caja de pandora" muy peligrosa puesto que un error del programador puede derivar en resultados inesperados de todo tipo. Veámoslo.



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

Conversión automática de tipos (type coercion)

► Observa:

"hola" se convierte automáticamente a boolean con valor true

```
JavaScript ▾  
if ("hola" && 3 < 10) {  
  console.log("Cierto");  
}
```

Console

"Cierto"

- Como vemos JS hace "conversiones automáticas" de los tipos de una expresión "forzándolos" para que todos se adapten a un tipo de dato.
- Estas conversiones automáticas se pensaron para "hacer la vida más fácil" al programador y pretendían solucionar cosas como esta:

JavaScript ▾

```
let edadActual = "10";  
edadPasada = edadActual - 1;  
console.log("Tienes "+edadActual+" años y el año pasado tenías "+edadPasada);
```

Console

"Tienes 10 años y el año pasado tenías 9"

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- Sin embargo, también nos puede dar errores inesperados como este:

JavaScript ▾

```
let edadActual = "10";  
edadNueva = edadActual + 1;  
console.log("Tienes "+edadActual+" años y el próximo año tendrás "+edadNueva);
```

Console

"Tienes 10 años y el próximo año tendrás 101"

JS convierte automáticamente el valor 1 a string para poder concatenarlo con "10". ¡MAL!

- Veamos algunos ejemplos extraños más:

"123" + 4 se evaluará como **"1234"**

"123" - 4 se evaluará como **119**

"1" - "1" se evaluará como **0**

34 + true se evaluará como **35**

2 * false se evaluará como **0**

-true se evaluará como **-1**

"true" + true se evaluará como **"truetrue"**

22

Las reglas que aplica JS para realizar estas conversiones automáticas las estudiaréis en el 2º curso

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

Comparando valores con == y ===

- ▶ JS sigue sorprendiéndonos:

```
JavaScript ▾  
if ("1" == 1) {  
    console.log("Son iguales");  
}
```

Console

"Son iguales"



En Java esto
provocaría
un error
sintáctico

- ▶ Con el operador == primero se realiza una **conversión automática de tipos** y después se comparan los valores resultantes.
- ▶ Este comportamiento facilita que se produzcan **errores difíciles de detectar**. Hoy en día se considera una mala práctica usar == y !=
- ▶ Más ejemplos de resultados inesperados:

Expresiones con == y !=
if ("123" == 123) devuelve <i>true</i>
if (0 == false) devuelve <i>true</i>
if ("" == false) devuelve <i>true</i>



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- ▶ Para solucionarlo, JS proporciona dos operadores más: `===` y `!==`
- ▶ Estos operadores **no hacen la conversión automática de tipos** y devuelven `false` siempre que los tipos ambos lados de la comparación no coincidan.
- ▶ Se considera buena práctica usar los operadores `===` y `!==`.
- ▶ Por cierto, las cadenas no se comparan con el método *equals* como en Java. También se compararan usando `===` y `!==`.
Observa:

JavaScript ▾	Console
<pre>let a = "hola"; let b = "hola"; if (a === b) { console.log("Iguales"); }</pre>	<pre>"Iguales" ></pre>

Con `===` se arreglan los problemas anteriores

Expresiones con <code>===</code> y <code>!==</code>
if ("123" === 123) devuelve <i>false</i>
if (0 === false) devuelve <i>false</i>
if ("" === false) devuelve <i>false</i>



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

Entrada por teclado

- Observa el siguiente trozo de código:
- Con **prompt** pedimos al navegador que muestre una ventana en la que puede teclear el usuario.
- El resultado tecleado se guarda en la variable **nombre** y se muestra en la consola:

JavaScript ▾

```
let nombre = prompt("Dime tu nombre: ");  
console.log("Te llamas "+nombre)
```

Una página insertada en esta dice

Dime tu nombre:

Pepe

Aceptar

Cancelar

Console

"Te llamas Pepe"

>

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- Si pinchamos en el botón "Cancelar" obtendríamos:

Console

```
"Te llamas null"
```



- También podemos poner un parámetro más que actúa como valor por defecto en la ventana:

JavaScript ▾

```
let nombre = prompt("Dime tu nombre: ", "Tu nombre aquí");  
console.log("Te llamas "+nombre);
```

Una página insertada en esta dice

Dime tu nombre:

Tu nombre aquí

Aceptar

Cancelar

Realiza el ejercicio 3
del boletín de
problemas

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- Observa también lo siguiente:

JavaScript ▾	Console
<pre>let edad = prompt("Dime tu edad: "); console.log("Hoy tienes "+edad+" años"); let edadNueva = edad + 1; console.log("El año que viene tendrás "+edadNueva+" años");</pre>	<pre>"Hoy tienes 32 años" "El año que viene tendrás 321 años"</pre>

- Esto se produce porque **prompt** devuelve un valor de tipo **string**. Para forzar el cambio de la cadena de texto a un número deberíamos usar **Number**, del siguiente modo:

JavaScript ▾	Console
<pre>let edad = <u>Number(prompt("Dime tu edad: "))</u>; console.log("Hoy tienes "+edad+" años"); let edadNueva = edad + 1; console.log("El año que viene tendrás "+edadNueva+" años");</pre>	<pre>"Hoy tienes 32 años" "El año que viene tendrás 33 años"</pre>

Number es un objeto "wrapper" que actúa como envoltorio del tipo primitivo number.

Realiza el ejercicio 4 del boletín de problemas

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

Funciones

- ▶ Las funciones son piezas de código aisladas que resuelven un problema concreto y que pueden ser invocadas desde otras partes del programa, pudiendo así reutilizarse. Son como los métodos de un objeto, pero sacados fuera del objeto.
- ▶ Las funciones deben ser "declaradas" antes de ser utilizadas. Observa la sintaxis de la declaración:

JavaScript ▾

```
// Declaración. Observa que los parámetros de entrada de la función
// se escriben directamente, sin "let" ni tipos ni nada.
function muestraSuma (num1, num2) {
  let suma = num1+num2;
  console.log ("La suma es "+suma);
}
```

Console



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- Sin embargo, el código anterior solo "declara" la función pero no la ejecuta. Para ello debemos "invocar o llamar" a la función dándole los parámetros de entrada que necesite. Observa:

JavaScript ▾

```
// Declaración. Observa que los parámetros de entrada de la función  
// se escriben directamente, sin "let" ni tipos ni nada.
```

```
function muestraSuma (num1, num2) {  
    let suma = num1+num2;  
    console.log ("La suma es "+suma);  
}
```

```
let a = 10, b = 20;
```

```
// Llamada a la función  
muestraSuma(a, b);
```

Console

"La suma es 30"

>

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- Dado que no se especifica el tipo de los parámetros de entrada, nuestro script puede hacer cosas raras dependiendo de los parámetros que se le pase. Observa:

JavaScript ▾

```
// Declaración. Observa que los parámetros de entrada de la función
// se escriben directamente, sin "let" ni tipos ni nada.
function muestraSuma (num1, num2) {
  let suma = num1+num2;
  console.log ("La suma es "+suma);
}
```

```
let a = "hola", b = "adios";
```

```
// Llamada a la función
muestraSuma(a, b);
muestraSuma("1", "2");
```

Console

"La suma es holaadios"

"La suma es 12"

>



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

- Si queremos que nuestra función calcule un dato y lo devuelva solo tenemos que añadir la instrucción **return**. Observa:

JavaScript ▾

```
function calculaSuma (num1, num2) {  
    var suma = num1+num2;  
    return suma;  
}  
  
let a = 10, b = 20;  
  
console.log("La suma de "+a+" y "+b+" es "+calculaSuma(a,b));
```

Console

"La suma de 10 y 20 es 30"



- También podríamos haber "recogido" el valor devuelto en una variable:

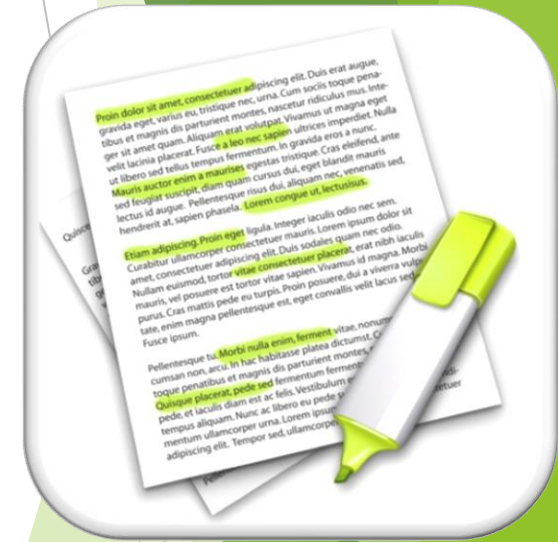
```
let a = 10, b = 20;  
  
let suma = calculaSuma(a,b);  
console.log("La suma de "+a+" y "+b+" es "+suma);
```

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

4 - Elementos básicos del lenguaje

Resumen de los elementos básicos del lenguaje:

- ▶ Las variables se declaran con **let** y su tipo se deduce de la asignación.
- ▶ El tipo de las variables puede cambiar.
- ▶ Las variables que no tienen un valor están marcadas como "undefined"
- ▶ La conversión automática de tipos puede jugar malas pasadas.
- ▶ Comparar con **===** y **!==** en vez de con **==** y **!=**
- ▶ Usaremos **prompt** para pedir datos al usuario. OJO que siempre devuelve una cadena. Si queremos pasarlo a número debemos hacer la conversión usando **Number(...)**
- ▶ Las funciones son como métodos fuera de un objeto. Hay que declararlas antes de poder llamarlas. Los parámetros de entrada no tienen tipo (puede haber comportamientos "raros")

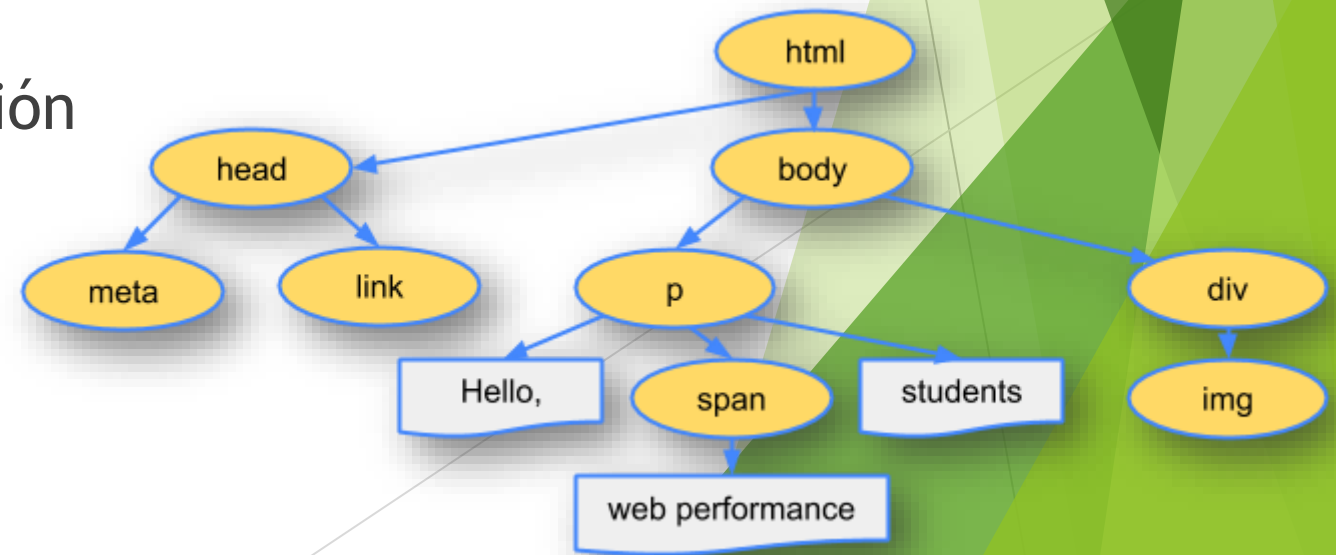


UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

5 - Mezclando JS con HTML

5 - Mezclando JS con HTML

- ▶ Cuando un navegador recibe una página web de un servidor tiene que hacer muuuchas cosas hasta poder "pintarla" en la ventana. Una de las tareas más importantes es **construir el DOM** (Document Object Model), es decir, el modelo de objetos del documento recibido.
- ▶ El DOM es una estructura de datos en forma de árbol donde cada nodo es un objeto que representa un elemento del documento HTML.
- ▶ Cada objeto contiene toda la información asociada al elemento HTML, es decir:
 - ▶ Textos
 - ▶ Propiedades CSS
 - ▶ Otros nodos...



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

5 - Mezclando JS con HTML

- ▶ Desde JS podremos manipular el DOM de una página web pudiendo añadir, borrar o modificar elementos del árbol y también reaccionar ante eventos. De este modo podemos dinamizar y enriquecer la experiencia del usuario con menús desplegables, con validaciones automática de datos, cambios dinámicos en la estética de la web, adaptar nuestra página a la dimensión de la pantalla de distintos dispositivos...
- ▶ Para ello debemos poder **asociar nuestro código JS** con el HTML para que puedan interactuar. Tenemos dos formas de hacerlo:
 - ▶ **Incluir el código JS directamente** en el documento HTML con la etiqueta `<script>`
 - ▶ **Incluir el código JS indirectamente** en el documento HTML haciendo referencia a un **fichero externo**.
- ▶ Veamos cada una de ellas.




HTML



JS

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

5 - Mezclando JS con HTML

- ▶ Incluir el código JS directamente
- ▶ Con etiquetas `<script>` podemos insertar código JS en nuestro documento HTML.
- ▶ Observa el código de al lado 
- ▶ Pueden aparecer tantas etiquetas `<script>` como se desee pero éstas solo pueden ubicarse:
 - ▶ Dentro del `<head>`
 - ▶ Dentro del `<body>`

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Incrustando JS en HTML</title>
</head>
<body>
  <h1>Incrustando JS con HTML</h1>
  <p>Párrafo escrito con HTML</p>
  <script>
    document.write("<p>Párrafo escrito desde JS</p>");
    console.log("Esto va a la consola de JS");
  </script>
</body>
</html>
```

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

5 - Mezclando JS con HTML

- ▶ En un navegador, el código anterior se vería así. Como vemos, el código JS se ejecuta exactamente en el lugar del documento HTML en el que se ha insertado.

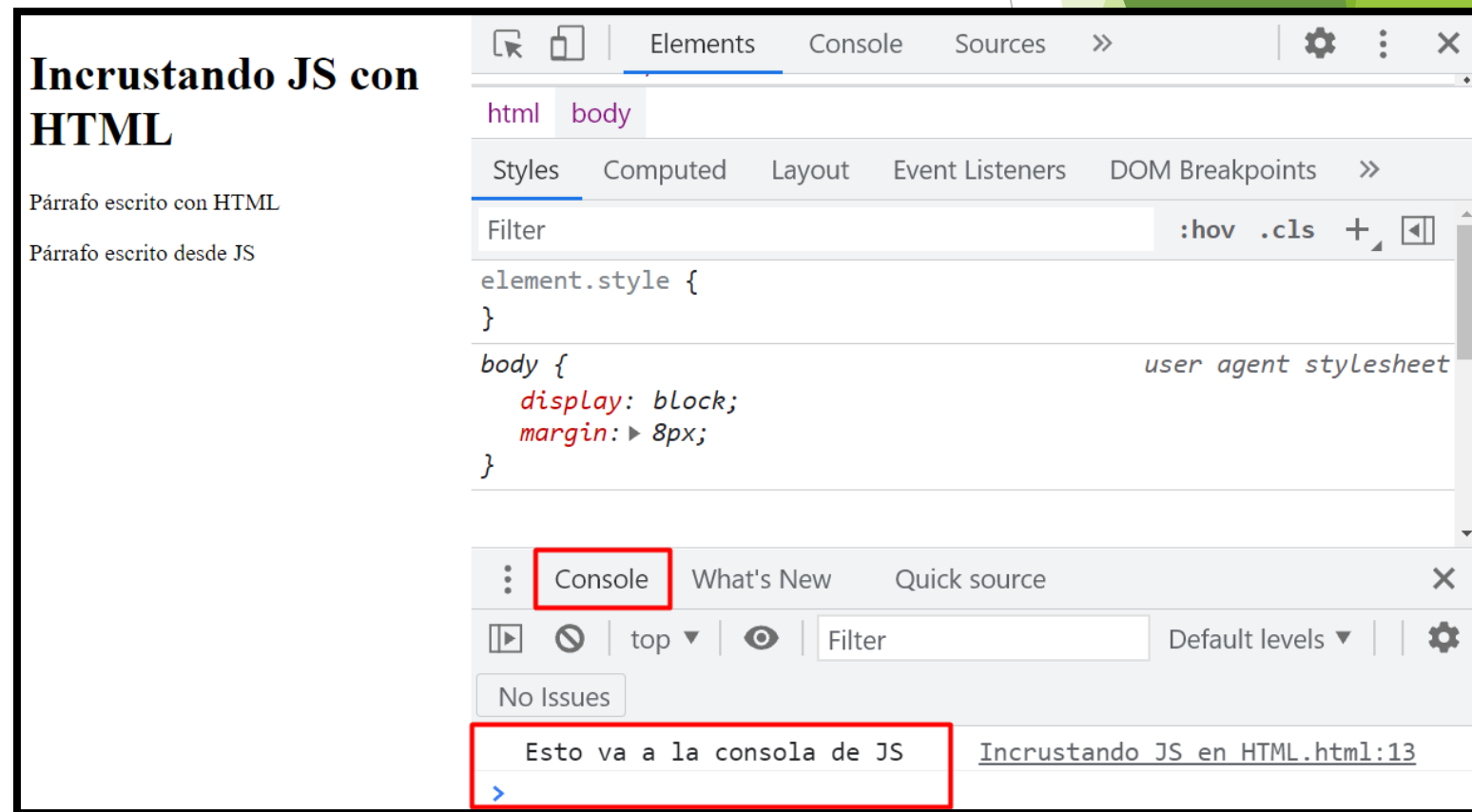


Incrustando JS con HTML

Párrafo escrito con HTML

Párrafo escrito desde JS

- ▶ Además, nuestro navegador tiene una consola de JS integrada que se muestra pulsando F12 (en el caso de las DevTools de Chrome)



UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

5 - Mezclando JS con HTML

- ▶ Incluir el código JS indirectamente
- ▶ En este caso la etiqueta `<script>` lleva un atributo `src` en el que se indica la ruta y el fichero que contiene el script.
- ▶ Cuando el navegador se encuentra esta etiqueta, busca el fichero asociado y después copia el código sobre el documento HTML y lo ejecuta.

OJO: como se hace una COPIA del código del fichero enlazado, la sintaxis de "autoclausura" NO FUNCIONA `<script src="..." />`

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>Enlazando JS en HTML</title>
</head>
<body>
  <h1>Enlazando JS con HTML</h1>
  <p>Párrafo escrito con HTML</p>
  <script src="script.js"></script>
</body>
</html>
```

```
<script>
< JS script.js X
G: > Mi unidad > EDUCACIÓN > Ciclo DAW > LENG. MARCAS Y SGI > Temario
1  document.write("<p>Párrafo escrito desde JS</p>");
2  console.log("Esto va a la consola de JS");
```


UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

5 - Mezclando JS con HTML

- ▶ El resultado de la ejecución del documento anterior es idéntico al caso anterior. De hecho, se está haciendo una COPIA del contenido del fichero JS de modo que el documento HTML resultante queda exactamente igual que cuando incluíamos el código directamente.
- ▶ Este mecanismo **acelera la carga de los sitios web en el navegador** ya que los ficheros de JS se quedan guardados en la **memoria caché** y, si hay varias páginas de un mismo sitio web que utilizan los mismos ficheros de script, entonces solo habrá que descargarlos una vez.
- ▶ Hay distintas políticas de ubicación de las etiquetas `<script>`:
 - ▶ Si se ponen al comienzo del documento HTML entonces la visualización de la página se retrasará pero tendremos toda la funcionalidad JS disponible desde el primer momento.
 - ▶ Si se ponen al final del documento HTML entonces la visualización de la página será más rápida pero puede que la funcionalidad no esté disponible desde el primer momento.

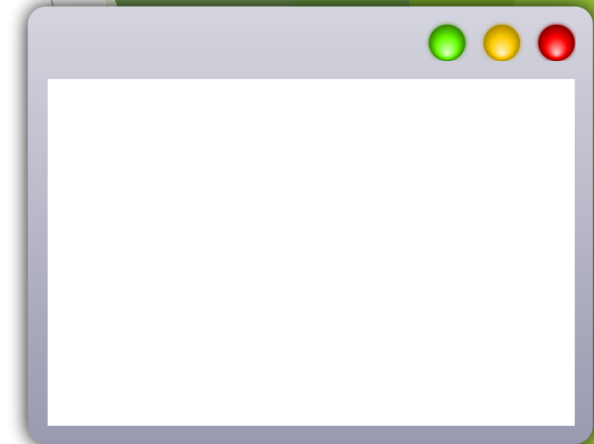
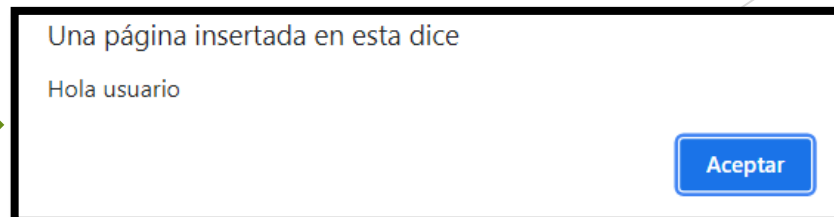


UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

5 - Mezclando JS con HTML

- ▶ Cuando JS se usa en un navegador se crea un crea automáticamente un objeto llamado **window** y que representa a la ventana del navegador:
 - ▶ Este objeto contiene las siguientes propiedades interesantes:
 - ▶ El objeto **document** que contiene el DOM de nuestro documento HTML y nos proporciona un montón de métodos útiles para manejarlo. El método **write** de este objeto nos permitirá añadir información en el DOM.
 - ▶ El objeto **console** que nos permitirá escribir en el consola JS del navegador.
 - ▶ Además, **window** proporciona dos métodos de uso habitual:
 - ▶ **prompt** para pedir información al usuario (ya lo estudiamos)
 - ▶ **alert** que nos permite mostrar una ventana de información al usuario del siguiente modo:

```
<script>  
  alert("Hola usuario");  
</script>
```



Realiza los
ejercicios del 5 al 9
del boletín de
problemas

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

6 - Introducción a la manipulación web con JS

- ▶ Ya hemos visto cómo escribir directamente en el documento HTML desde JS usando **document.write(...)**
- ▶ Ahora vamos a aprender a seleccionar un elemento de HTML para hacerle cambios. Observa el código de al lado.

HTML ▾

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
  <p id="demo"></p>
```

```
<script>
  let parrafo = document.getElementById("demo");
  parrafo.innerText="Hola mundo";
</script>
```

```
</body>
</html>
```

Output

Hola mundo

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- ▶ Con `document.getElementById("demo")` estamos seleccionando el elemento `<p id="demo">` de nuestro documento HTML. Y con la propiedad `innerText` cambiamos el contenido de texto de dicho elemento.
- ▶ Si el elemento buscado no se encuentra entonces no se cambia nada en el documento HTML, pero nos sale un error en la consola de JS:

<div>HTML ▾</div> <pre><!DOCTYPE html> <html> <head> </head> <body> <p id="demo"></p> <script> let parrafo = document.getElementById("noExiste"); parrafo.innerHTML = "Hello World!" </script> </body> </html></pre>	<div>Console Clear</div> <div>"error"</div> <div>"TypeError: Cannot set properties of null (setting 'innerHTML') at <anonymous>:3:23 at https://static.jsbin.com/js/prod/runner-4.1.8.min.js:1:13924 at https://static.jsbin.com/js/prod/runner-4.1.8.min.js:1:10866"</div> <div>></div>	<div>Output</div>
---	--	-------------------

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- También podemos ahorrarnos la variable *párrafo* y escribirlo todo en una línea:

```
<script>
  document.getElementById("demo").innerText="Hola mundo";
</script>
```

- La propiedad `innerHTML` es similar a la anterior pero ahora nos devuelve las etiquetas HTML además del texto. Observa la diferencia:

```
HTML ▾
<!DOCTYPE html>
<html>
<head>
</head>
<body>
  <p id="demo">Esto es un parrafo con <b>negrita</b></p>

  <script>
    let varInnerText = document.getElementById("demo").innerText;
    let varInnerHTML = document.getElementById("demo").innerHTML;
    console.log("varInnerText = "+varInnerText);
    console.log("varInnerHTML = "+varInnerHTML);
  </script>
</body>
</html>
```

Console

```
"varInnerText = Esto es un parrafo con negrita"
"varInnerHTML = Esto es un parrafo con <b>negrita</b>"
```

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- OJO: si usamos ambas propiedades para modificar el contenido de un elemento los resultados serán distintos. Observa la diferencia:

HTML ▾

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <p id="demoText"></p>
  <p id="demoHTML"></p>

  <script>
    document.getElementById("demoText").innerText = "<b>Parrafo negrita</b>";
    document.getElementById("demoHTML").innerHTML = "<b>Parrafo negrita</b>";
  </script>
</body>
</html>
```

Output 727px

`Parrafo negrita`

Parrafo negrita

Realiza los ejercicios del 10 al 11 del
boletín de problemas

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- También podemos consultar y modificar los **atributos** de un elemento HTML. Observa:

```
<!DOCTYPE html>
<html>
  <head></head>
<body>
  

  <script>
    let imagen = document.getElementById("imagen");
    console.log("Imagen antes del cambio = "+imagen.src);
    imagen.src = "http://sabemos.es/wp-content/uploads/2016/11/vodafone.png";
    console.log("Imagen después del cambio = "+imagen.src);
  </script>
</body>
</html>
```



Console

```
"Imagen antes del cambio = https://null.jsbin.com/sin-imagen.jpg"
"Imagen después del cambio = http://sabemos.es/wp-content/uploads/2016/11/vodafone.png"
```


UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- Y también podemos consultar y modificar las **propiedades CSS** de un elemento HTML. Observa:

HTML ▾

```
<html>
<head></head>
<body>
  <h1 id="titular">Hola a todos</h1>

  <script>
    document.getElementById("titular").style.color = "blue";
    document.getElementById("titular").style.margin = "50px";
  </script>

</body>
</html>
```

Output

Hola a todos

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- Y también podemos modificar la propiedad class que usamos con CSS de un elemento HTML. Observa:

```
<html>
<head></head>
<body>
  <style>
    .mi-titulo-rojo {
      color: red;
      margin: 50px;
    }
    .mi-titulo-azul {
      color: blue;
      text-align: center;
    }
  </style>
  <h1 id="titular">Hola a todos</h1>

  <script>
    document.getElementById("titular").className = "mi-titulo-azul";
  </script>

</body>
</html>
```

Hola a todos

Realiza el ejercicio 12 del boletín de problemas

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

Seleccionando varios elementos

- ▶ JS también permite seleccionar varios elementos del árbol que posean alguna característica común y procesarlos en un bucle.
- ▶ Observa:

Lorem Ipsum

Lorem ipsum dolor sit amet consectetur adipiscing elit, turpis donec tortor vulputate arcu scelerisque fermentum hac, cum sed magnis at elementum litora.

Eget fusce mus rutrum dictum cursus vivamus augue metus iaculis, lobortis mauris leo orci luctus mollis morbi molestie sociis, tortor placerat diam elementum magnis consequat justo eu.

Ligula nostra cubilia sociis lacus sodales mattis elementum arcu, eleifend imperdiet habitasse suscipit curabitur nisi mauris, molestie proin ultricies bibendum blandit rhoncus massa.

```
<html>
<head></head>
<body>
  <h1 id="titular">Lorem Ipsum</h1>
  <p>Lorem ipsum dolor sit amet consectetur adipiscing elit, tur
  <p>Eget fusce mus rutrum dictum cursus vivamus augue metus iac
  <p>Ligula nostra cubilia sociis lacus sodales mattis elementum

  <script>
    let parrafos = document.getElementsByTagName("p");
    // parrafos es un array de JavaScript

    for (i = 0; i < parrafos.length; i++) {
      let parrafo = parrafos[i];
      parrafo.style.margin = "15px";
      parrafo.style.color = "blue";
    }
  </script>
</body>
</html>
```

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- ▶ Como vemos **getElementsByTagName** nos devuelve un array con todos los elementos del DOM que se correspondan con la etiqueta indicada. Si no encuentra ninguno nos devolverá **null**.
- ▶ Si queremos ser un poco más elegantes y evitar que la consola de JS del navegador se llene de errores podemos preguntar si el array obtenido es nulo antes de entrar en el bucle que lo recorre.

```
if (parrafos !==null) {  
  for (i = 0; i < parrafos.length; i++) {  
    let parrafo = parrafos[i];  
    parrafo.style.margin = "15px";  
    parrafo.style.color = "blue";  
  }  
}
```

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- También podemos usar `getElementsByClassName` que nos devolverá un array con todos los elementos del DOM que tengan un atributo de clase determinado (o null si no encuentra ninguno).

Output

Artículo 1

Artículo 2

Artículo 3

```
<html>
<head></head>
<body>
  <h1 class="titular">Artículo 1</h1>
  <h1 class="titular">Artículo 2</h1>
  <h1 class="titular">Artículo 3</h1>

  <script>
    let titulares = document.getElementsByClassName("titular");

    if (titulares !==null) {
      for (i = 0; i < titulares.length; i++) {
        let titular = titulares[i];
        titular.style.margin = "15px";
        titular.style.color = "blue";
      }
    }
  </script>
</body>
</html>
```

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

6 - Introducción a la manipulación web con JS

- También podemos modificar el ejercicio anterior para que se añada un botón, que al presionarlo, provoque el cambio de formato. Observa:

```
<html>
<head></head>
<body>
  <h1 class="titular">Artículo 1</h1>
  <h1 class="titular">Artículo 2</h1>
  <h1 class="titular">Artículo 3</h1>

  <button onclick="formatear()">Cambiar formato</button>

  <script>
    function formatear() {
      let titulares = document.getElementsByClassName("titular");

      for (i = 0; i < titulares.length; i++) {
        let titular = titulares[i];
        titular.style.margin = "15px";
        titular.style.color = "blue";
      }
    }
  </script>
</body>
</html>
```

Asociamos la acción de hacer click en el botón con el código de la función "formatear" que hay más abajo

Artículo 1
Artículo 2
Artículo 3

Cambiar formato

Realiza los ejercicios 13 y 14 del boletín de problemas

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB

Cierre de la unidad

Cierre de la unidad

- ▶ En este tema solo hemos **arañado la superficie** de JS como lenguaje para manipular páginas web.
- ▶ Si quieres profundizar más te recomiendo el siguiente material:
 - ▶ Cursos online
 - ▶ W3Schools: <https://www.w3schools.com/js/default.asp>
 - ▶ Uniwebsidad: <https://uniwebsidad.com/libros/javascript>
 - ▶ Libro 'Eloquent JavaScript' de Marijn Haverbeke:
 - ▶ Versión online gratuita en inglés: <https://eloquentjavascript.net/>
 - ▶ Versión traducida al español casi al completo: <https://eloquentjs-es.thedojo.mx/>
 - ▶ Cuando ya te sientas cómodo/a con el lenguaje, intenta leer el siguiente libro: 'JavaScript: The Good Parts'



Saber
más

UD X - INTRODUCCIÓN A JS COMO LENGUAJE DE LA WEB



The End